SE 3XA3: Test Plan
Jumbled Words

Team 08, Shunbill Jumble
Shesan Balachandran, balacs1
Bill Nguyen, nguyew3
Muneeb Arshad, arsham14

March 5, 2021

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
March 04 2021	0.0	Initial Draft

1 General Information

1.1 Purpose

The purpose of this test plan is to outline and show how our Jumbled Words product will be tested. This document is important since proper/detailed testing of any software product is necessary in order to ensure that the product will function as intended according to the Software Requirements Specification (SRS). The previous document which was the SRS was completed prior to this document in order to form the basis for the creation of these tests. The testing plan document will be used during the creation of the product/software in order to validate and check that the system that is being created will satisfy all the necessary requirements. Also, this document will be used when writing the necessary software tests and when testing the product overall in order to ensure everything is satisfied.

1.2 Scope

The test plan will provide the outline for testing the functionality for the modification of the Jumbled Word Game. The scope of testing will cover the front-end components and the accuracy of their interactions and the accuracy of the backend logic which includes the game itself, user state management and the overall business logic of the product. These three categories will have details testing to cover specific requirements and product functionalities.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
SRS	Software Requirement Specification
POC	Proof of Concept
FR	Functional Requirement
NFR	Non-Functional Requirement

1.4 Overview of Document

The following sections of this document will be outlining and describing how our product “Jumbled Words” will be tested. Section 2 will consist of the main/overall test plan that will outline and describe what the specific software is, who will test the software, what are tools/software that will be used to perform these tests and what the test schedule will be. For section 3, this section will describe what functional and non-functional test cases that will be created for every area of the product/system and will also provide a traceability matrix for these test cases which will trace them to the system requirements. Section 4 will provide what was presented during the POC demonstration and how what was demonstrated was tested. Section 5, will be a comparison from the existing implementation to what is going to be added and improved on for this product. Section 6, will outline how the unit tests will be created and these unit tests will test the individual methods and functionality of the product. Finally, section 7 is the appendix of this document, which will just provide any additional information such as symbolic parameters and whether there is a usability survey.

2 Plan

2.1 Software Description

The Jumbled Words Quiz In Python is a simple project for helping your kids grow in IQ. The project contains only the user side. The user can start the quiz by clicking on the start button. Also, you can choose the type of words, you want to solve in the quiz. The user can change jumbled words if they do not know the correct word. The players get points for guessing each word. The players can also access the leaderboard from the main menu to view player high scores. The game has different game modes and difficulty levels to challenge the player.

2.2 Test Team

The test team consists of all the members in Group 08, Shesan Balachandran, Bill Nguyen and Muneeb Arshad.

2.3 Automated Testing Approach

The automated testing approach is to develop test suites with automated testing frameworks and apply regression testing techniques as new features are developed. The automated testing shall be managed through a script that would run all the test suites and output the testing results. Managing all the testing through a script will allow the team to test using a standardized approach throughout the team but also improve testing speeds as well.

2.4 Testing Tools

We will be using the testing framework pytest to test the Python code. To test the statement coverage and the branch coverage we will be using coverage.py. Moreover to detect incorrect naming conventions of variables and functions, unused code/variables and to access the complexity of the code we will be using the vscode built-in static analysis tool for python called pylint.

2.5 Testing Schedule

All tasks that need to be accomplished are covered in the Gantt Chart provided in the project planning, which includes the allocation of time for specific tasks as well as the deadlines.

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 User Input

Screen Interactions

1. FR-SCRN-1

   Type: Functional, Manual

   Initial State: The start screen is displayed

   Input: Click start game button

Output: Game screen switches from the start screen to options screen

How test will be performed: The tester selects the start game button on the main menu and the test asserts if the user interface switches and displays the option screen

2. FR-SCRN-2

   Type: Functional, Manual

   Initial State: Main screen

   Input: Click Leaderboard Button

   Output: Leaderboard is loaded

   How test will be performed: The tester selects the Leaderboard button on the main menu and checks if the user interface switches and displays the leaderboard screen

3. FR-SCRN-3

   Type: Functional, Manual

   Initial State: Main screen

   Input: Quit Button

   Output: The application shuts down

   How test will be performed: The tester selects the Quit button on the main menu and the checks if the game application shuts down

4. FR-SCRN-4

   Type: Functional, Manual

   Initial State: Screen with back button

   Input: Click Back Button

   Output: The previous page is displayed

   How test will be performed: To ensure all the back buttons have proper functionality we have to test the back button on each page. The tester will click the back button on the options screen and check the user interface switches to the Main Menu. Similarly, the back button will be clicked on the difficulty options screen and game mode options screen to check whether they go back to the previous page.

Gamplay Interactions

1. FR-GMPY-1

   Type: Functional, Manual

   Initial State: Empty input string

   Input: Type the word using keyboard

   Output: The typed word is displayed on screen

   How test will be performed: An expected text is typed through the keyboard and the actual typed word is asserted with the expected word

2. FR-GMPY-2

   Type: Functional, Manual

   Initial State: Null state for the feedback notification

   Input: Click submit button

   Output: The result of the submission (correct/incorrect) will be displayed in the notification.

   How test will be performed: For a sample question, the submit button is clicked and is checked with whether the state for the feedback notification is changed from null to showing state.

3. FR-SCRN-3

   Type: Functional, Manual

   Initial State: A question is displayed on screen

   Input: Click next question button

   Output: If the correct answer was submitted already, change screens else remain on same screen

How test will be performed: For a sample question, the submit button is clicked with a correct and incorrect answer, and is checked whether the screen changes for the answers respectively

Settings Menu

1. FR-STNG-1

   Type: Functional, Manual

   Initial State: Difficulty Level Option Screen

   Input: Click Easy Difficulty

   Output: Timer corresponds to the difficulty level

   How test will be performed: The tester selects the Easy difficult button on the main menu then selects the random category and checks if the User Interface displays the game play screen and the timer corresponds to the difficulty level selected

2. FR-STNG-2

   Type: Functional,Manual

   Initial State:Difficulty Level Option Screen

   Input:Click Medium Difficulty

   Output: Timer corresponds to the difficulty level

   How the test will be performed: The tester selects the Medium difficult button on the main then selects the random category menu and checks if the User Interface displays the game play screen and the timer corresponds to the difficulty level selected.

3. FR-STNG-3

   Type: Functional, Manual.

   Initial State:Difficulty Level Option Screen

   Input:Click Hard Difficulty

   Output: Timer corresponds to the difficulty level

How the test will be performed: The tester selects the Hard difficult button on the main menu then selects the random category and checks if the User Interface displays the game play screen and the timer corresponds to the difficulty level selected

4. FR-STNG-4

   Type: Functional, Dynamic.

   Initial State:Sample username

   Input:Enter Username

   Output: Username exists in the json file

   How the test will be performed: A unit test will be performed by entering different inputs(string, integers, special characters) as username and check whether the username gets stored in the username json file

5. FR-STNG-5

   Type: Functional, Manual

   Initial State:Username screen

   Input:Enter Username

   Output: User Interface switches to next screen

   How the test will be performed: The tester types in the username and checks whether the screen displays the typed username. The user selects the enter button and ensures the user interface switches to the game mode option screen.

6. FR-STNG-6

   Type: Functional, Manual

   Initial State:Categories option screen

   Input:Select Category

   Output: User interface switches to game play

How the test will be performed: The tester is on the category options screen and can see all the available categories. Then a random category is selected and the user interface switches to another screen and displays the game play. The tester verifies the sample words matches the category selected.

7. FR-STNG-7

   Type: Functional, Manual

   Initial State:Game Mode Option Screen

   Input:Select Game Mode

   Output: User interface switches display

   How the test will be performed: The tester is on the game mode options screen and can see all the available game modes. Then a specific game mode(practice, timed) is selected and the user interface switches to another screen and displays the difficult level option screen. The tester will proceed to the game play screen by selecting all other game settings and verify the game play has a timer when the timed mode is selected and no timer exists when practice mode is selected.

3.1.2 Game Logic

1. FR-GMLG-1

   Type: Functional, Static, Automatic

   Initial State: A sample question and a timer is selected

   Input: A correct answer is submitted at a specific time

   Output: Certain amounts of points are calculated for the answer given at the time

   How the test will be performed: Unit tests will be developed to check whether points are calculated accurately based on the time in the timer.

2. FR-GMLG-2

   Type: Functional, Static, Automatic

Initial State: A sample question is provided

Input: A correct answer and an incorrect is provided

Output: The result of the submission (correct/incorrect) will be displayed in the notification

How the test will be performed: Unit tests will be developed to check whether the sample words do indeed match the expected words

3. FR-GMLG-3

   Type: Functional, Static, Automatic

   Initial State: A sample question is provided

   Input: No input is provided and the time runs out

   Output: The next button will become active to go to the next question

   How the test will be performed: Unit tests will be developed to check whether the state of next button clickability is matched with timeouts

3.1.3 Leaderboard Testing

1. FR-LEAD-1

   Type: Functional, Dynamic, Manual

   Initial State: The leaderboard has no high scores

   Input: a new score is accomplished

   Output: The leaderboard will display only the new score on the leaderboard

   How the test will be performed: We will clear the high scores of all the users manually before starting the game and complete the game, which will generate a new score. Then we will make sure that the new score is the only score on the leaderboard and is at the top.

2. FR-LEAD-2

   Type: Structural, Dynamic, Manual

   Initial State: The leaderboard is full (has 10 entries)

Input: a new score is accomplished that is lower than all of the scores displayed in the leaderboard

Output: The leaderboard will not be updated

How the test will be performed: We will manually fill the leaderboard until it is full, then we will produce a score that is lower than all of the scores displayed in the leaderboard and ensure that the leaderboard did not change

3. FR-LEAD-3

   Type: Functional, Dynamic, Manual

   Initial State: The leaderboard is full (has 10 entries)

   Input: a new score is accomplished that is higher than any of the scores displayed in the leaderboard

   Output: The leaderboard will be updated, removing the lowest score in the leaderboard and adding the new score in the corresponding order (from greatest to least)

   How the test will be performed: We will manually fill the leaderboard until it is full, then we will produce a score that is higher than any of the scores displayed in the leaderboard and ensure that the leaderboard removed the lowest score that is displayed in the leaderboard and added the new score. Also the leaderboard should display the scores from greatest to least

4. FR-LEAD-4

   Type: Functional, Dynamic, Manual

   Initial State: The leaderboard has 3 entries

   Input: a unique new score is accomplished that is less than the current lowest score on the leaderboard

   Output: The leaderboard will be updated, and should have 4 entries, with newest score being placed last and should be displayed from greatest to least

   How the test will be performed: We will manually fill 3 entries in the leaderboard, then we will produce a new unique score that is higher than any of the 3 entries in the leaderboard. Then we will check if leaderboard is displayed from greatest to least

5. FR-LEAD-5

   Type: Functional, Dynamic, Manual

   Initial State: The leaderboard has 3 entries

   Input: a unique new score is accomplished that is higher than any of the current scores on the leaderboard

   Output: The leaderboard will be updated, and should have 4 entries, with the leaderboard being displayed from greatest to least

   How the test will be performed: We will manually fill 3 entries in the leaderboard, then we will produce a new unique score that is lower than the 3 entries in the leaderboard. Then we will check if the new unique score is below the other 3 scores

3.2 Tests for Nonfunctional Requirements

3.2.1 Usability and Humanity Requirements

1. NFR-UH-1

   Type: Structural, Dynamic, Manual

   Initial State: Game is installed on a local device, and launched using the executable and goes to the Main Menu of the game

   Input/Condition: User is expected to navigate through the Main Menu page without assistance

   Output/Result: Majority of the users are able to access, start and view different parts of the game via the Main Menu page intuitively and without assistance

   How test will be performed: We will have a selected group of users that have no background of how the product works will be given the product in the Main Menu state and will be asked to perform different tasks from the Main Menu page, such as starting the game, accessing the leaderboard etc. and if these users can do these tasks in a short amount of time then this test shall be considered successful

2. NFR-UH-2

   Type: Structural, Static, Manual

   Initial State: The 3 members/developers of the product have a great understanding of the product and implementation

   Input: The members will each think of 2-3 questions about usability to test each other, and gage every members opinion on the usability of the product

   Output: The members will answer yes or no to these questions and if majority of the members answer yes (2 out of 3 members), this question will be deemed correct

   How test will be performed: Each member will think of 2-3 questions related to the usability of the product and they will answer the questions themselves, and the other members will answer the questions and if majority of the questions, the members agree that this is satisfied then the test will be a success

3.2.2 Performance

1. NFR-PER-1

   Type: Non Functional, Manual

   Initial State: The game will be made available with all active functionality

   Input: Every clickable action is attempted several times and full game is played in different modes and difficulty level

   Output:The time taken for the buttons to respond and the next page to load is recorded. If any of the response is less than 2seconds the tests are failed

   How the test will be performed: The tester of the game will ensure the game is loaded on the computer. Then the tester will perform all clickable events on the menu options and game play with different game modes. The other tester will watch and record the response time for each event and record the time. If the response is greater than 2seconds then test will be marked as failed.

3.2.3 Operational And Environmental Requirements

1. NFR-OE-1

   Type: Non Functional, Manual

   Initial State: The game must be available online to download

   Input: The game is downloaded on different windows version and macOS

   Output: The game is supported on different operating systems

   How the test will be performed: The tester will manually download the game available online on different operating systems (Windows version 20H2 and macOS version 10.15.7) and verify the game is supported with all features

2. NFR-OE-2

   Type: Non Functional, Manual

   Initial State: The game must be available online to download

   Input: The game is downloaded on computer with 1Gb ram and 1Gb disk space

   Output: The game is supported on computer with 1Gb ram and 1Gb disk space

   How the test will be performed: The tester will manually download the game available online on a computer with 1Gb ram and 1Gb disk space and verify the game is supported with all features

3.3 Traceability Between Test Cases and Requirements

Traceability for the functional requirements and the test cases

FR01: FR-STNG-4
FR02: FR-STNG-7
FR03: FR-STNG-1,FR-STNG-2,FR-STNG-3
FR04: FR-STNG-6
FR05: FR-GMLG-1

FR06: FR-GMLG-2
FR07: FR-GMLG-1
FR08: FR-SCRN-3
FR09: FR-STNG-5
FR10: FR-GMLG-3
FR11: FR-LEAD-1,FR-LEAD-2,FR-LEAD-3,FR-LEAD-4,FR-LEAD-5
FR12: FR-SCRN-1

Traceability for the non-functional requirements and the test cases

NFR2: NFR-UH-1
NFR12: FR-SCRN-1
NFR3 : NFR-PER-1
NFR4.1: NFR-OE-1
NFR4.2: NFR-OE-2

4 Tests for Proof of Concept

4.1 Best Score Testing

1. POC-1

   Type: Functional, Manual

   Initial State: Game is started with settings of choice

   Input: The game is played

   Output: The final score is the highest score

   How the test will be performed: A sample playthrough is made where the answers are already given prehand so the final score will result in the highest score possible

4.2 Quit

1. POC-2

   Type: Functional, Manual

Initial State: Game is started with settings of choice

Input: The quit button is selected

Output: The ends and the state is idempotent

How the test will be performed: A sample play-through is made but the game is closed mid way to validate the states remain idempotent

5 Comparison to Existing Implementation

The existing implementation had no testing included in the project. Therefore a whole set of test cases were developed to validate the existing software as well as the new features. Extensive testing is done on the front-end components as that is what is primarily kept from the existing product.

6 Unit Testing Plan

6.1 Unit testing of internal functions

The python testing framework pytest will be used for the unit testing of internal functions. For functions that directly return values, assert statements will be used to directly check the output. For functions that do not directly return values or for the testing of the UI elements, separate state variables will be used and modified by the interactions, which will then be checked through the assert statements. For example, to test if the back button works, a separate state variable can be used to validate whether the back button changed the state since it is difficult to test that directly.

6.2 Unit testing of output files

There are no output files for our product.